Received January 30, 2020, accepted February 12, 2020, date of publication February 14, 2020, date of current version February 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973924

Few-Shot Learning Based Balanced Distribution Adaptation for Heterogeneous Defect Prediction

AILI WANG, YUTONG ZHANG, HAIBIN WU, KAIYUAN JIANG, AND MINHUI WANG
Higher Education Key Laboratory for Measuring and Control Technology and Instrumentations of Heilongjiang, Harbin University of Science and Technology, Harbin 150080, China

Corresponding author: Haibin Wu (woo@hrbust.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant NSFC-61671190.

ABSTRACT Heterogeneous defect prediction (HDP) aims to predict the defect tendency of modules in one project using heterogeneous data collected from other projects. It sufficiently incorporates the two characteristics of the defect prediction data: (1) datasets could have different metrics and distribution, and (2) data could be highly imbalanced. In this paper, we propose a few-shot learning based balanced distribution adaptation (FSLBDA) approach for heterogeneous defect prediction, which takes into consideration the two characteristics of the defect prediction data. Class imbalance of the defect datasets can be solved with undersampling, but the scale of the training datasets will be smaller. Specifically, we first remove redundant metrics of datasets with extreme gradient boosting. Then, we reduce the data difference between the source domain and the target domain with the balanced distribution adaptation. It considers the marginal distribution and the probability of conditional distribution differences and adaptively assigns different weights to them. Finally, we use adaptive boosting to relieve the influence caused by the size of the training dataset is smaller, which can improve the accuracy of the defect prediction model. We conduct experiments on 17 projects from 4 datasets using 3 indicators (i.e., AUC, G-mean, F-measure). Compared to three classic approaches, the experimental results show that FSLBDA can effectively improve the prediction performance.

INDEX TERMS Heterogeneous defect prediction, balanced distribution adaptation, few-shot learning, class imbalance.

I. INTRODUCTION

With the availability of massive storage capabilities, high speed Internet, and the advent of Internet of Things devices, modern software systems are growing in both size and complexity [1]. Software Defect Prediction (SDP) can accurately find defects in the early stages of software development. It focuses on identifying defect tendencies in software modules and helps researchers allocate limited resources to modules with high probability of containing defects. SDP can solve the problem of insufficient energy of developers and limited development cycle, on the other hand, it can effectively improve the quality of software.

Cross-project defect prediction (CPDP) utilizes the existing historical data of other projects to construct a prediction model, which does not require sufficient historical data of the project to be predicted. However, the source project is required to have common metrics as the target project. However, the programming languages and application domains of different projects are often different, the corresponding metrics are different [2]. Heterogeneous Defect Prediction (HDP) can reduce significant difference, whose prediction effect is independent of whether the two projects have common metrics or not. Li et al. proposed a new cost sensitive transfer kernel canonical correlation analysis (CTKCCA) approach for HDP, which made the data distributions of source and target projects much more similar in the nonlinear feature space [3]. Li et al. not only made better use of two projects but also alleviated the class imbalance problem by setting different misclassification costs for different samples [4]. Li et al. proposed a multi-source selection based manifold discriminant alignment (MSMDA) approach. A sparse representation based double obfuscation algorithm is designed and applied to HDP [5].

The researchers used Domain Adaptation (DA) to reduce significant difference of data, no longer requiring that the source project (source domain) has the same metrics and distribution as the target project (target domain). There are three kinds of DA. Distribution adaptation focuses on the data distribution of source domain and target domain. Feature

The associate editor coordinating the review of this manuscript and approving it for publication was Shiqiang Wang.

selection focuses on the shared metrics of source domain and target domain. Subspace learning is concerned with subspaces shared by the source domain and the target domain.

Feature selection and subspace learning are often used in the field of heterogeneous software defect prediction. The researchers used the feature selection method to select shared metrics from the source domain and target domain, and constructed a unified model [6], [2]. Yu et al. grouped the original metrics with spectral clustering according to the correlation of metrics, and employed Relief F algorithm to compute the relevance between each metric with respect to the number of faults, and selected the most relevant metrics from each resulted cluster [7]. Subspace learning transforms the source domain and the target domain into the same subspace. He et al. used clustering and data selection to select the subset highly relevant to the target domain to learn the similar distribution between the two domains [8]. According to the distribution curve of metrics, Yu et al. matched the heterogeneous metrics and aligned the source domain with the target domain [9]. Wen et al. adopted feature selection, which combined with transfer component analysis (TCA+) for spatial transformation, and obtained accurate prediction results [10]. Chen et al. proposed a heterogeneous data orienting multi-view transfer learning for software defect prediction, which can achieve different dimensions and granularities metrics to automatically learn labels through neural network models [11]. Chen et al. proposed a collective training mechanism for defect prediction (CTDP), which made the distributions of source and target projects similar to each other by transfer learning [12]. Liu et al. proposed a spatial-neighborhood manifold learning (SNML) framework for data analysis, which used the spatial-neighborhood information to construct the adjacency graph [13]. However, feature selection is limited by two reasons: whether there are common metrics that affect the classification results in the source domain and the target domain, or whether there is a greater correlation between the important metrics of the source domain and the target domain. Subspace learning can reduce data drift during data mapping, but there are still different marginal distribution and conditional distribution in source domain and target domain, which affect the decision result.

Distribution adaptation mainly considers the probability of marginal distribution and the probability of conditional distribution. Some methods [14], [15] consider only one of these aspects. Long et al. proposed the joint distribution adaptation method (JDA) to match the marginal distribution and conditional distribution between different domains [16]. Others extended JDA by adding structural consistency [17], domain-invariant clustering [18], and target selection [19]. These methods tend to ignore the importance of two different distributions and simply add them up. However, when there are large differences between two distributions, these methods cannot assess the importance of each distribution and may not be well generalized in most cases. Existing distribution adaptation methods generally assume that two distributions are equally important. However, this assumption does not hold. For example, when there is a big difference between the source domain and the target domain data, marginal distribution adaptation is more important. Conditional probability distribution adaptation is more important when the datasets of source domain and target domain have higher similarity. Wang et al. proposed a balanced distribution adaptation method (BDA) [20], which can dynamically measure the different effects of marginal distribution and conditional distribution, rather than simply give them the same weight.

In particular, there is class imbalance in the datasets. In order to obtain the model with better classification effect, undersampling is carried out on non-defective samples. However, the quantity of training data is small, which easily leads to under-fitting of the prediction results of the model. Ensemble learning can complete the few-shot learning and obtain a complex learning model by constructing and combining multiple base classifiers. It does not require additional parameters and avoids costly off-line training sessions [22]. Li et al. developed an ensemble multiple kernel correlation alignment (EMKCA) predictor, which combined the advantage of multiple kernel learning with domain adaptation techniques [23]. Li et al. proposed a novel Two-Stage Ensemble Learning (TSEL) approach to HDP, which learned multiple different EMKCA predictors and used average ensemble to combine them together. [24]. Boosting in ensemble learning can use a small amount of data to iteratively generate multiple learners with weak generalization performance and construct a strong ensemble classification model [21].

This paper proposes a heterogeneous defect prediction method, denoted as FSLBDA, which combines ensemble learning with domain adaptation. It addresses the fact that there are no or fewer common metrics between two projects. The feature selection of the source domain is realized by extreme gradient boosting (XGBoost). Domain adaption reduces the data distribution difference between source domain and target domain. Undersampling was used to solve the class imbalance, but a small number of defective samples resulted in a small balance training set. Adaptive boosting (AdaBoost) iteratively updates sample weight and reduces the deviation of the classification surface. The main contributions of this article are as follows:

(1) AdaBoost is used to realize the few-shot learning to prevent under-fitting of the prediction model obtained from the small data set (a balanced data set is obtained from undersampling).

(2) BDA can dynamically measure the importance of marginal distribution and conditional distribution and realize adaptive distribution adaptation.

The remainder of the paper is organized as follows. Section II presents the architecture of the proposed approach. Section III is the theory of feature selection and classification. Section IV describes the main principle of balanced distribution adaptation. Section V analyzes and discusses the experiments results. Finally, conclusions are drawn in Section VI.

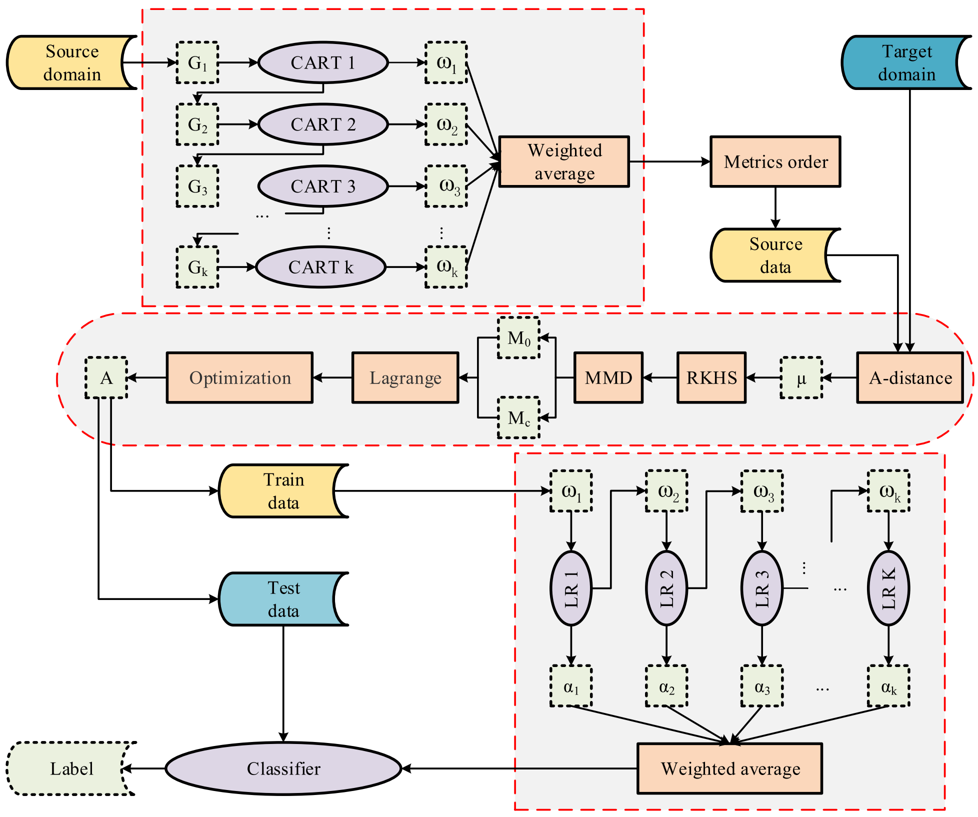


FIGURE 1. **The overall architecture of FSLBDA.**

II. PROPOSED APPROACH

As shown in Figure 1, the proposed approach framework of this paper is mainly divided into three parts. Firstly, XGBoost selects the metrics in the source domain. Secondly, BDA minimizes data distribution differences between source and target domains. Thirdly, AdaBoost classifies the samples of the target domain.

The first step is to select metrics in the source domain. Defective samples and non-defective samples are selected with the ratio of 1:1 to construct a balanced data set. XGBoost is adopted, and the gradient lifting algorithm is used to continuously reduce the loss of the previously generated decision tree. It updates the weight G_i of the samples and generate a new decision tree. The weights of different metrics in all sub-leaves ω_j were weighted and averaged to determine the importance of the metrics. The complexity of the tree model is added to the objective function to avoid over-fitting. The first and second derivatives of the Taylor expansion of the objective function are applied to accelerate the optimization.

The second step is to measure the conditional distribution and marginal distribution of source domain and target domain dynamically to reduce data difference adaptively. We use A-distance to estimate the balance factor μ. A-distance is defined as establishing a linear classifier to distinguish hinge losses in two data fields. Maximum Mean Discrepancy (MMD) is used to calculate the difference between the two probability distributions for obtaining the matrix M_0 and M_c. The optimal transformation matrix A is obtained by using Lagrange multiplier optimization. The source domain and target domain data with similar distribution can be obtained by transformation matrix.

The third step is to use AdaBoost to predict the defect tendency of the target domain modules. Each training sample is initially assigned the same weight. If a sample has been accurately classified, its weight is reduced in constructing the next training set. Conversely, if a sample is not accurately classified, its weight is increased. Then the sample sets with updated weight ω_i are used to train in the next classifier, and the training process proceed iteratively. After the training process of each weak classifier was completed, the weight α_i of the weak classifier with low classification error rate was increased, while the weight of the weak classifier with high classification error rate was decreased. Weighted average was used to determine the predicted results of defect samples.

III. FEATURE SELECTION AND CLASSIFICATION

Boosting's evolutionary methods in ensemble learning include XGBoost and AdaBoost. This paper adopts the above ideas to achieve the feature selection of source domain and defect tendency prediction of target domain respectively. Boosting generates a strong learner with nearly perfect performance by increasing the number of iteration times of the weak leaner. The weak learner means that the classification effect is only slightly better than the random guess effect.

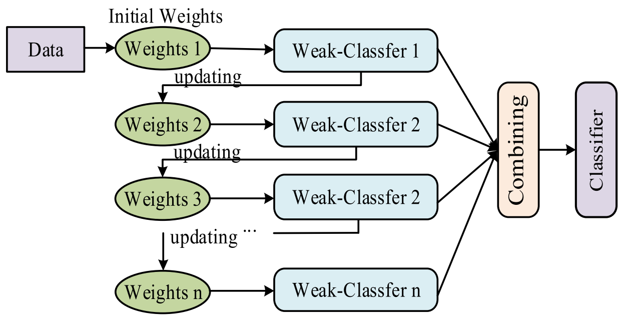


FIGURE 2. **Algorithm flow chart of Boosting.**

In practice, it is easier to get a weak learner than a strong learner. Each classifier generated after the first classifier is to learn from the samples that were not correctly classified in the previous time, which can effectively reduce the deviation of the model. As shown in Figure 2, Boosting repeatedly runs a weak learning to deal with different weight of training data, then the weak learners generated each time are combined into a composite strong learner.

XGBoost quantifies the importance of each metric using the characteristics of the Classification and Regression Tree (CART) to select the partitioning points. In order to minimize the cost of the segmented tree, the metric with the highest gain is selected for segmentation until the maximum depth. The gradient lifting algorithm is used to continuously reduce the loss of the previously generated decision tree, which minimizes the objective function and ensures the reliability of the final decision. The objective function considers the complexity of the tree model to avoid over-fitting. The loss function is expanded by Taylor expansion, and the first and second derivatives are used to accelerate the optimization.

The objective function to be minimized is as follows. l is a differentiable convex loss function to predict $\hat{y}_i$ and target y_i differences. Ω is penalty term of the model complexity, which can help smooth the final weights of learning and avoid over-fitting. $\hat{y}_{i(k)}$ is the prediction of the i-th sample at the k-th iteration. $q(x_i)$ is the structure function of each tree that maps an example to the corresponding leaf index. The objective function greedily adds p_k p_k. Each p_k corresponds to an independent tree structure q and leaf weights ω. T is the number of leaf nodes, and $\|\omega\|$ is the magnitude of leaf node vector. γ represents the parameter for adjusting the shading of a node, and λ represents the L2 regularization coefficient.

$$L_k = l\left(y_i, \hat{y}_{(k-1)} + p_k(x_i)\right) + \Omega(p_k) \tag{1}$$

$$\Omega(p_k) = \gamma T + \frac{1}{2}\lambda \|\omega\|^2 \tag{2}$$

$$p_k(x_i) = \omega_{q(x_i)} \tag{3}$$

Second-Order approximation optimizes the model quickly, where $g_i = \partial_{\hat{y}_{i(k-1)}} l(y_i, \hat{y}_{i(k-1)})$, $h_i = \partial^2_{\hat{y}_{i(k-1)}} l(y_i, \hat{y}_{i(k-1)})$ are the first and the second order gradient statistics of the loss function. I_j is the sample set of leaf j. The objective function after removing the constant term can be expressed as follows.

$$\begin{aligned} L_k &= \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_{i(k-1)}\right) + g_i p_k + \frac{1}{2} h_i p_k^2\right] + \Omega(p_k) \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)\omega_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right)\omega_j^2\right] + \gamma T \end{aligned} \tag{4}$$

The weight ω_j of each leaf in each tree is obtained, which is used to calculate the metric importance finally.

$$\omega_j = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \tag{5}$$

AdaBoost changes the weight of training data, which is the probability distribution of samples. Its idea is to focus on the samples that are wrongly classified, reduce the sample weight of the last round of correct classification, and improve the sample weight of those wrongly classified. AdaBoost uses the method of weighted majority voting, which increases the weight of weak classifiers with small classification error rate and reduces the weight of weak classifiers with large classification error rate. The weight of the sample is mainly used for the weak classifier to find the decision point with the smallest classification error, and then the weight of the weak classifier is calculated with this minimum error. The larger weight of the classifier has the greater voice in the final decision.

During the process of training the weak classifier, the objective function can be optimized as:

$$L = \operatorname{argmin}_{\beta,f} \sum_{i=1}^{1} \exp\left(-y_i\left(F_{j-1}(x_i) + \beta f(x_i)\right)\right) \tag{6}$$

β is the weight coefficient and f represents the prediction function of the weak classifier. $F(x_i)$ represents the function of the strong classifier constituted by iteration. The exponential loss function is used instead of the mean square error loss function because the latter is not effective for classification applications. The first part of the exponential function represents the loss function of the existing strong classifier on a single training sample. The latter part is the loss function of the current weak classifier to the training sample. The objective function can be simplified as:

$$(\beta_j, f_j) = \operatorname{argmin}_{\beta,f} \sum_{i=1}^{1} \omega_i^j \exp\left(-y_i \beta f(x_i)\right) \tag{7}$$

$$\omega_i^j = \exp\left(-y_i F_{j-1}(x_i)\right) \tag{8}$$

ω_i^j is the sample weight, which is only related to the strong classifier obtained in the previous iteration, and has nothing to do with the current weak classifier. This optimization problem can be solved in two steps. First, β is regarded as a constant. Since the values of y_i and $f(x_i)$ can only be $+1$ or -1, they must be equal to minimize the objective function. Therefore, the optimal solution is:

$$f_j = \operatorname{argmin}_f \sum_{i=1}^{1} \omega_i^j I\left(y_i \neq f(x_i)\right) \tag{9}$$

where I is the indicator function, whose value is 0 or 1 according to the conditions in brackets. The optimal solution is the classifier which can make the weighted error rate of the sample minimum. The optimization objective can be expressed as:

$$F(\beta) = \left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{1} \omega_i^j I\left(y_i \neq f(x_i)\right) + e^{\beta} \sum_{i=1}^{1} \omega_i^j \tag{10}$$

The derivative of the function at the extreme point is 0, so the optimal solution of β is obtained:

$$\beta = \frac{1}{2}\log\frac{1 - err_j}{err_j} \tag{11}$$

err_j is the weighted error rate of the weak classifier to a training set:

$$err_j = \frac{\sum_{i=1}^{N} \omega_i^j I\left(y_i \neq f(x_i)\right)}{\sum_{i=1}^{N} \omega_i^j} \tag{12}$$

The updating formula of sample weight in iteration is written as follows:

$$\omega_i^{j+1} = \omega_i^j \times e^{-y_j \beta_j f(x_j)} \tag{13}$$

IV. BALANCED DISTRIBUTION ADAPTATION

In software defect prediction problems, labeled source domain and unlabeled target domain often differ in both marginal and conditional distributions. Figure 3 demonstrates the importance of matching both marginal and conditional distributions for domain adaptation. If the two distributions are treated equally, they cannot take full advantage of each other's importance. When two domains are very dissimilar (Figure 3(a) $\rightarrow$ (b)), the marginal distribution is more important to align. When the marginal distributions are close (Figure 3(a) $\rightarrow$ (c)), the conditional distribution should be given more weight. BDA can adaptively adjust the importance of two distributions to achieve the better performance. Giving a labeled source domain $\left\{X_{s_i}, y_{s_i}\right\}_{i=1}^{n}$, an unlabeled target domain $\left\{X_{t_i}\right\}_{i=1}^{m}$, their marginal distribution $P_s \neq P_t$ and conditional distributions $Q_s \neq Q_t$, BDA aims to learn the labels y_t of the target domain D_t using the source domain D_s. Domain Adaptation often adaptively minimize the marginal and conditional distribution discrepancy between domains. Specifically, this refers to minimizing the distance:

$$d(D_s, D_t) \approx d(P_s, P_t) + d(Q_s, Q_t) \tag{14}$$

BDA exploits a balance factor μ to leverage the different importance of distributions where $\mu \in [0, 1]$. When $\mu \rightarrow 0$, it means the datasets are more dissimilar, so the marginal distribution is more dominant; when $\mu \rightarrow 1$, it reveals the datasets are similar, so the conditional distribution adaptation is more important. Therefore, the balance factor μ can adaptively utilize the importance of each distribution and lead to good results.

Since the target domain D_t has no label, the conditional distribution P_t cannot be directly obtained. Instead, we use the class conditional distribution to approximate P_t. To calculate the class conditional distribution, D_t is predicted using the base classifier trained on the source domain D_s, soft labels are obtained and constantly corrected. In order to compute the discrepancies between the marginal distribution and the conditional distribution, we used MMD to estimate the discrepancies between the two distributions. $d(D_s, D_t)$ can be represented as:

$$d(D_s, D_t) \approx (1-\mu)\left\|\frac{1}{n}\sum_{i=1}^{n} X_{S_i} - \frac{1}{m}\sum_{j=1}^{m} X_{t_j}\right\|_H^2 + \mu\sum_{c=1}^{C}\left\|\frac{1}{n_c}\sum_{X_{S_i}\in D_s^{(c)}} X_{S_i} - \frac{1}{m_c}\sum_{X_{t_j}\in D_t^{(c)}} X_{t_j}\right\|_H^2 \tag{15}$$

H denotes the reproducing kernel Hilbert space, and $c \in \{1, 2\}$ refers to the distinct class labels. n, m represent the number of samples in the source domain and the target domain respectively. D_s^c and D_t^c represent samples belonging to class c in the source and the target domain respectively. The corresponding samples is $n_c = \left|D_s^c\right|, m_c = \left|D_t^c\right|$. By further using the matrix operation rules, the above formula is formalized as follows:

$$\min \operatorname{tr}\left(A^T X\left((1-\mu) M_0 + \mu\sum_{c=1}^{2} M_c\right) X_T A\right) + \lambda \|A\|_F^2 \tag{16}$$

$$A^T X H X^T A = I, \quad 0 \leq \mu \leq 1 \tag{17}$$

The former term is used to adapt marginal distribution and conditional distribution with balance factor μ, and the latter term is regularization. λ denotes regularized parameter for Frobenius norm $\|\bullet\|_F^2$. There are two main influence factors. One is the transformed data $\left(A^T X\right)$ which holds the internal properties of the original data. The second is the value range of balance factor. X is the input matrix composed of X_s and X_t, A is the transformation matrix, $I \in R^{(n+m)\times(n+m)}$

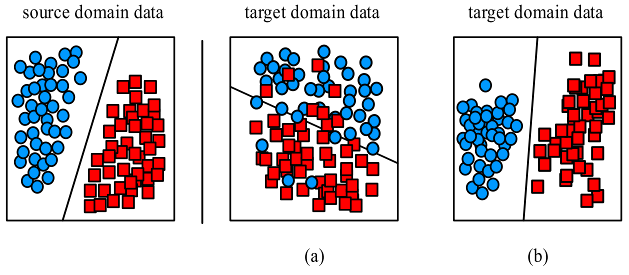


FIGURE 3. As shown in figure (a), the marginal distribution is preferred for the data of the target domain, and the conditional distribution is preferred for the data in figure (b).

is the identity matrix, $H = I - (1/n) I$ is the centering matrix. M_0 and M_c are MMD matrices. By multipliers $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_d)$, the Lagrange function as follows:

$$L = \operatorname{tr}\left(A^T X\left((1-\mu) M_0 + \mu \sum_{c=1}^{2} M_c\right) X_T A\right) + \lambda \|A\|_F^2 + \operatorname{tr}\left(\left(1 - A^T X H X^T A\right) \Phi\right) \quad (18)$$

Set the derivative $\frac{\partial L}{\partial A} = 0$, the optimization can be derived as a generalized Eigen decomposition problem to find out d minimum eigenvectors, and then the optimal transformation matrix A is obtained. Source domain and target domain with the least discrepancy can be obtained by transformation matrix.

$$\left(X\left((1-\mu) M_0 + \mu \sum_{c=1}^{2} M_c\right) X^T + \lambda I\right) = XHX^T A \Phi \quad (19)$$

According to the difference of marginal distribution, A-distance between the source domain and the target domain is calculated, denoted as A_M. For conditional distribution differences, we first cluster the target domain into C classes, and then calculate A-distance of the data from the same class in the two domains. The average of A-distance between all categories is denoted as A_C. Then, can be estimated as $\mu \approx A_C / (A_C + A_M)$.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. DATASETS DESCRIPTION

Two projects of the Relink dataset (Safe and Zxing) are used as the source domain data in the following experiments [25]. The Relink dataset was collected using the Understand tool (https://scitools.com) by Wu et al., with 26 metrics that measure code complexity. A total of 15 projects were selected from the data sets of AEEEM, NASA and SOFTLAB as the predicted target domain. The AEEEM dataset was collected by D'Ambros et al. Each AEEEM dataset consists of 61 metrics including object-oriented (OO) metrics, previous-defect metrics, entropy metrics of change and code, and churn-of-the source code metrics [26]. ReLink and AEEEM have no common metrics. The static code measures involved of NASA dataset include lines of code, software complexity, and software readability, all of which are closely related to software quality. Shepperd et al. found conflicts and inconsistencies in the NASA dataset and cleaned it up, which is the cleaned-up version used in the experiments [27]. The SOFTLAB dataset comes from the Turkish software company. ReLink and NASA have three common metrics, including lines of code, blank lines, and comment lines, while others have no common metrics.

Precisely, Table 1 summarizes the 17 datasets utilized in this paper. We can see that the imbalance ratio varies from 1.51 (only slightly imbalanced) to 12.44 (highly imbalanced). We also considered datasets with diversity in the number of instances; the smallest dataset has 36 samples, while the largest dataset contains 1862 samples.

B. EXPERIMENTAL RESULTS

Experiments used the scikit-learn under Linux as the backend. Python the multi-paradigm programming language with rich data science packages has been selected. The information of hardware is CPU: Intel®Core^{TM}i7-9750H, Video card: NVIDIA Geforce RTX 2060.

There are four possible output results for any sample in the target domain after the defect prediction model: when a sample containing defects is predicted to be a defect sample, it is denoted as TP (true positive); when a sample without defects is predicted as a defect sample, it is denoted as FP (false positive); when a sample containing defects is predicted to be a non-defective sample, it is denoted as FN (false negative); when a sample without defects is predicted as non-defective sample, it is denoted as TN (true negative). Based on the above possible output results, Precision, Recall, TNR, G-mean, and F1-measure can be defined.

TABLE 1. Details of projects used in experiment.

Project	Source	Description	Metrics	Instances	Non-Defects	Defects	Defective (%)	IR
Safe	Relink	Security	26	56	34	22	39.29%	1.55
Zxing		Bar-code reader library	26	399	281	118	29.57%	2.38
PC3	NASA	Flight Software for Each Orbiting Satellite	37	1077	943	134	12.44%	7.04
PC4		Flight Software for Each Orbiting Satellite	37	1458	1280	178	12.21%	7.19
MW1		A Zero Gravity Experiment about Combustion	37	253	226	27	10.67%	8.37
cm1		Spacecraft instrument	37	327	285	42	12.84%	6.79
PC1		Flight software for earth orbiting satellite	21	1109	1032	77	6.94%	13.40
ML	AEEEM	Task management	61	1862	1617	245	13.16%	6.60
PDE		IDE development	61	1497	1288	209	13.96%	6.16
EQ		OSGi Framework	61	324	195	129	39.81%	1.51
JDT		IDE Development	61	997	791	206	20.66%	3.84
LC		Text Search Engine Library	61	691	627	64	9.26%	9.80
AR1	SOFTLAB	Embedded controller for white-goods	29	121	112	9	74.00%	12.44
AR3		Embedded Controller of The Washing Machine	29	63	55	8	12.70%	6.88
AR4		Embedded Controller of The Dishwasher	29	107	87	20	18.69%	4.35
AR5		Embedded Controller of The Refrigerator	29	36	28	8	22.22%	3.50
AR6		Embedded controller for white goods	29	102	87	15	14.70%	5.80

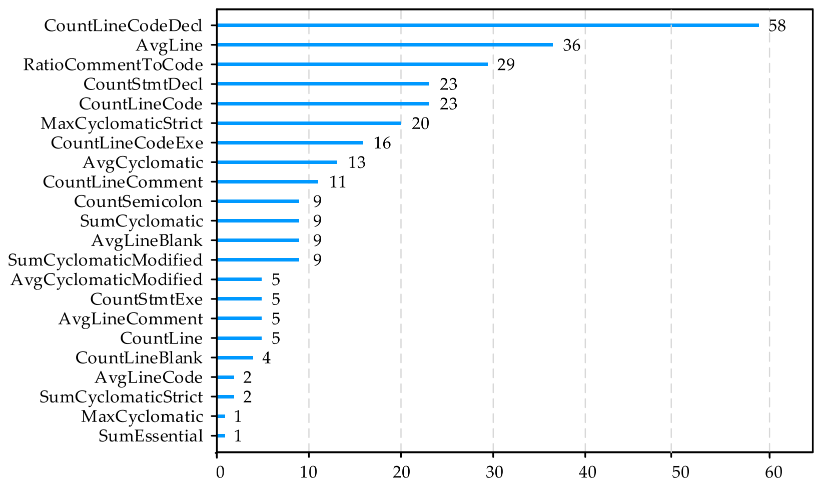


FIGURE 4. The metric ranking scores for Safe.

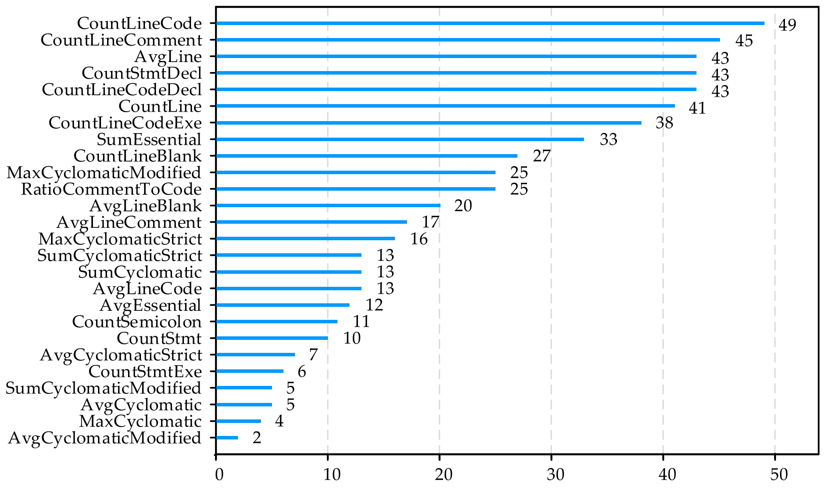


FIGURE 5. The metric ranking scores for Zxing.

Precision is the percentage of all samples that are predicted to be defective that actually contain defects.

$$\text{Precision} = \text{TP}/\left(\text{TP} + \text{FP}\right) \tag{20}$$

Recall is the percentage of samples that are correctly predicted to be a defective sample.

$$\text{Recall} = \text{TP}/\left(\text{TP} + \text{FN}\right) \tag{21}$$

TNR is the percentage of samples that are predicted to be a non-defective sample.

$$\text{TNR} = \text{TN}/\left(\text{TN} + \text{FP}\right) \tag{22}$$

G-mean can be used to evaluate the model performance of imbalanced data.

$$\text{G-mean} = \sqrt{\text{TNR} \times \text{Recall}} \tag{23}$$

F1-measure comprehensively considers the Precision and Recall.

$$\text{F1} - \text{measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

In general, when Precision is high, Recall is often low. Recall is high, Precision is often on the low side. However, it is not enough to use accuracy or recall only as evaluation index. F1-measure is obtained through the harmonic mean calculation of Precision and Recall. In addition, the area under the working characteristic curve (AUC) was not influenced by threshold value and the class imbalance. In view of this, this paper uses AUC, G-mean and F1-measure to evaluate the performance of different approaches. Three classical approaches of heterogeneous software defect prediction are conducted to evaluate the performance of FSLBDA in heterogeneous defect prediction, including TCA+ [28], CCA+ [29] and KCCA+ [9].

Figure 4 and Figure 5 are the metric ranking scores of Safe and Zxing. The score is obtained by XGBoost considering the complexity of the tree. We get rid of irrelevant metrics based on the score we get. The higher the score of a metric, the more meaningful it is for classification. We can find that the importance of the same metric differs between the two projects. The specific meaning of each metric can be queried

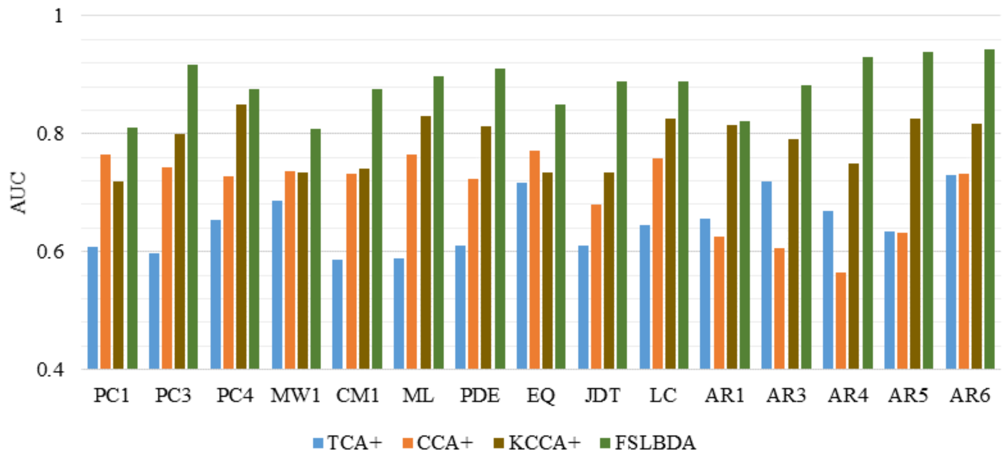


FIGURE 6. AUC of FSLBDA and other approaches on 15 tasks when Safe project as the source domain.

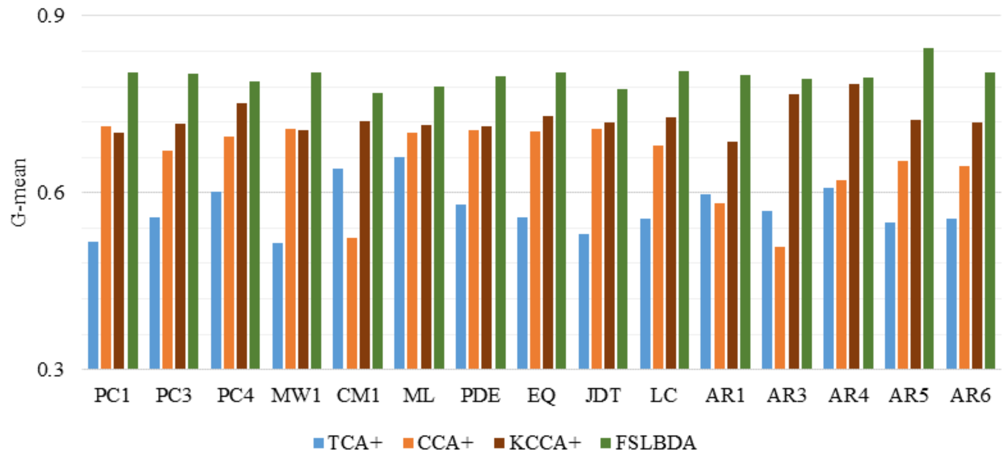


FIGURE 7. G-mean of FSLBDA and other approaches on 15 tasks when Safe project as the source domain.

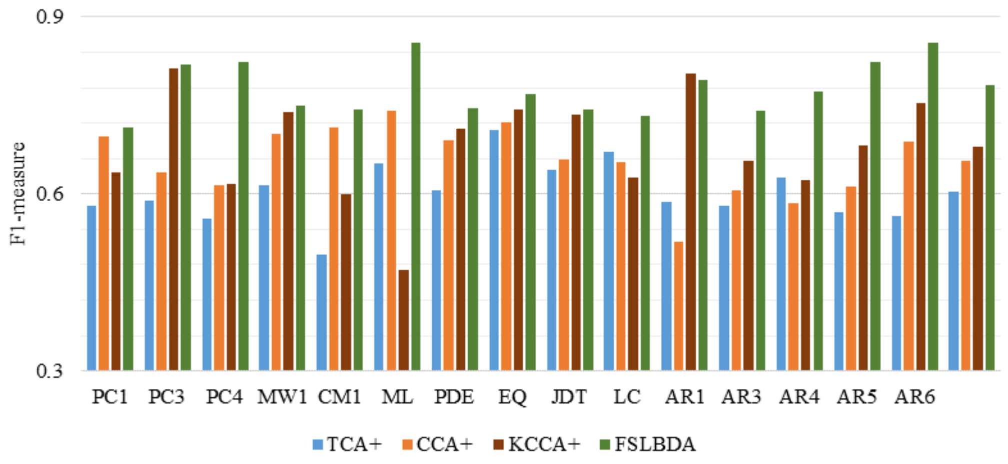


FIGURE 8. F1-measure of FSLBDA and other approaches on 15 tasks when Safe project as the source domain.

through Understand website. Additionally, the source domain training set for balanced distribution adaptation was obtained by undersampling, and the Safe project for 44 samples and the Zxing project for 236 samples were obtained respectively. The new projects with class balance are more favorable to predict the defect tendency of 15 target projects.

It can be seen from the statistical results in Table 2 when Safe is the data of the source domain, AUC of FSLBDA is

TABLE 2. Three indicator values of approaches when safe project as the source domain.

	AUC				G-mean				F1-measure			
Project	**TCA+**	**CCA+**	**KCCA+**	**FSLBDA**	**TCA+**	**CCA+**	**KCCA+**	**FSLBDA**	**TCA+**	**CCA+**	**KCCA+**	**FSLBDA**
PC1	0.608	0.764	0.720	0.810	0.567	0.722	0.723	0.804	0.580	0.698	0.636	0.712
PC3	0.598	0.743	0.799	0.917	0.561	0.680	0.704	0.789	0.589	0.636	0.812	0.819
PC4	0.654	0.728	0.850	0.875	0.511	0.692	0.716	0.766	0.559	0.614	0.616	0.823
MW1	0.687	0.737	0.734	0.808	0.589	0.697	0.740	0.804	0.614	0.702	0.739	0.750
CM1	0.587	0.733	0.740	0.875	0.514	0.510	0.712	0.770	0.498	0.713	0.600	0.743
ML	0.589	0.764	0.830	0.898	0.531	0.676	0.763	0.800	0.652	0.741	0.472	0.857
PDE	0.611	0.723	0.813	0.911	0.547	0.648	0.702	0.801	0.607	0.692	0.710	0.745
EQ	0.716	0.771	0.735	0.850	0.607	0.570	0.687	0.802	0.708	0.721	0.743	0.769
JDT	0.610	0.680	0.735	0.889	0.549	0.633	0.710	0.776	0.640	0.658	0.735	0.744
LC	0.646	0.758	0.826	0.889	0.637	0.682	0.786	0.806	0.672	0.655	0.627	0.732
AR1	0.655	0.625	0.814	0.821	0.549	0.518	0.745	0.777	0.587	0.519	0.803	0.793
AR3	0.719	0.606	0.791	0,882	0.606	0.529	0.749	0.792	0.579	0.607	0.657	0.740
AR4	0.669	0.565	0.749	0.930	0.460	0.519	0.704	0.746	0.628	0.585	0.624	0.773
AR5	0.634	0.633	0.825	0.938	0.559	0.538	0.744	0.736	0.570	0.613	0.683	0.824
AR6	0.731	0.733	0.816	0.944	0.515	0.656	0.690	0.761	0.562	0.688	0.753	0.857
mean	0.648	0.704	0.785	0.883	0.553	0.618	0.725	0.782	0.603	0.656	0.681	0.779

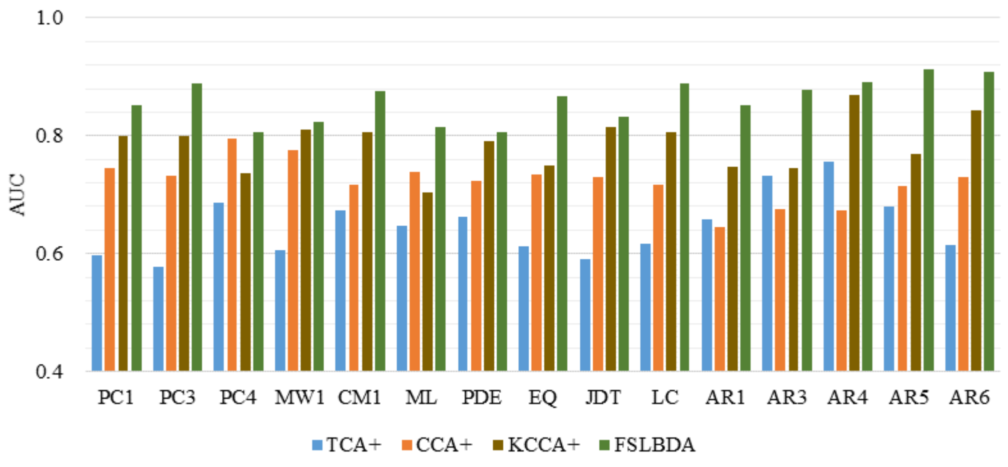


FIGURE 9. AUC of FSLBDA and other approaches on 15 tasks when Zxng project as the source domain.

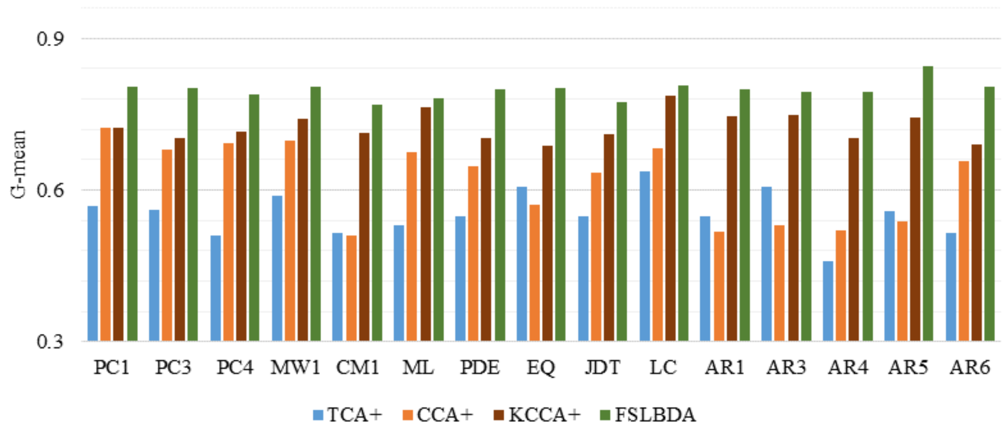


FIGURE 10. G-mean of FSLBDA and other approaches on 15 tasks when Zxng project as the source domain.

between 0.808 and 0.944, with an average value of 0.883. PC1, MW1 and AR1 projects are less than 0.85, and the rest are all above 0.85. The average AUC values of the other three classical approaches were 0.648, 0.704, and 0.785, respectively, and the improvement rates were 28.54%, 25.42%, and 12.48%. Except for the AR5 project, G-mean of all the other

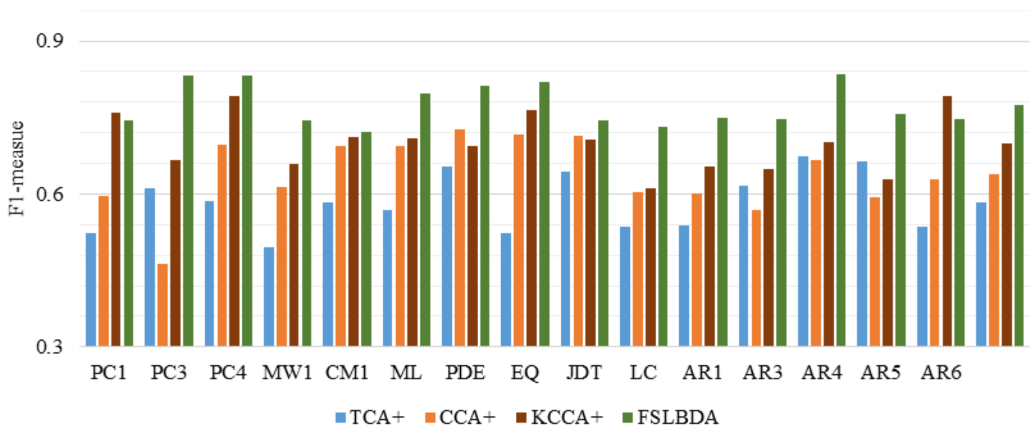


FIGURE 11. F1-measure of FSLBDA and other approaches on 15 tasks when Zxng project as the source domain.

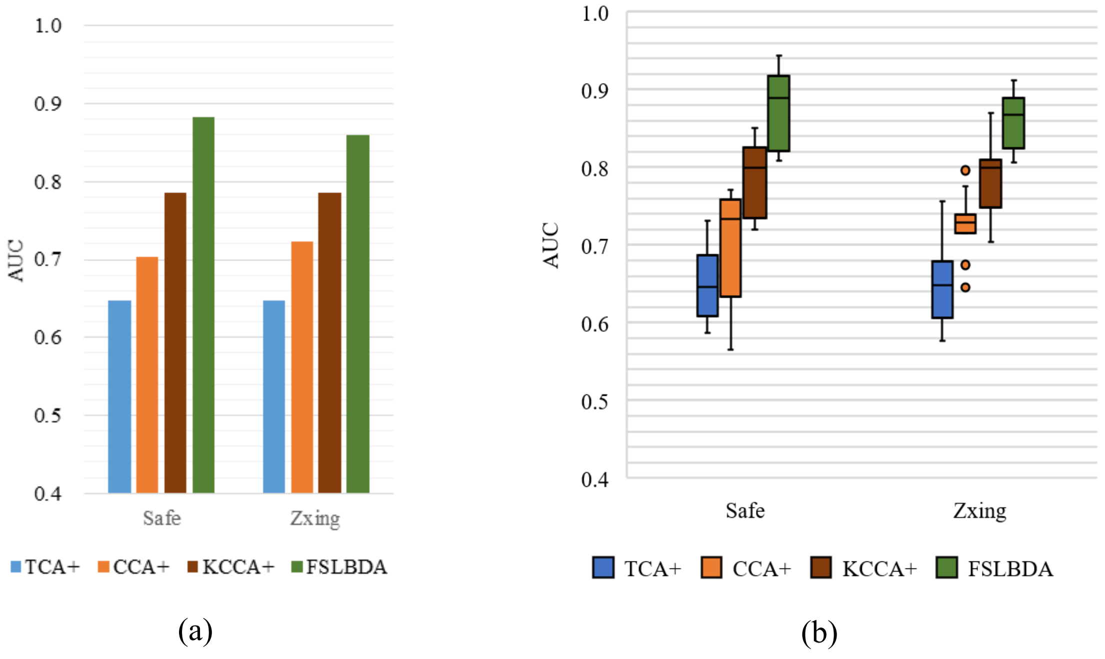


FIGURE 12. The AUC of different approaches. (a) is the histogram, and (b) is the boxplot.

projects are superior compared to classical approaches with an average of 0.782. F1-measure of FSLBDA is between 0.712 and 0.857, with a mean of 0.779.

It can be seen from Figure 6, 7, 8 that AUC, G-mean and F1-measure generated by FSLBDA for each target data set are mostly higher than the other three approaches, indicating that the performance of FSLBDA in correctly predicting defective and non-defective classes is better than others. The result shows that whether the source project and the target project have common metrics, the prediction effect has been improved.

When Zxing is the data of the source domain, the predicted results are shown in Table 3. It can be seen from the statistical results that the AUC of FSLBDA is between 0.806 and 0.912, with an average value of 0.860. Mean value of AUC for other three classic approaches are 0.647, 0.723, 0.786 respectively, improving 32.92%, 18.95%, 9.41%. In terms of G-mean, FSLBDA is superior to other classical approaches, with a mean of 0.798. The other classic approaches averages of 0.573, 0.655 and 0.726, respectively. G-mean of FSLBDA increased 0.225, 0.143 and 0.072, respectively. F1-measure is between 0.722 and 0.836, with a mean of 0.774. Except for PC1 and AR6 projects, FSLBDA is better than the other three classical approaches. As can be seen from Figure 9, 10, 11, the classification effect of FSLBDA is improved compared with the other three approaches. The result shows that FSLBDA is not only applicable to the case with common metrics, but also can be used to predict the defect tendency without common metrics.

For comprehensive assessment of the overall performance of the proposed approach in this paper, Figure12, 13, 14 shows the mean of AUC, G-mean and F1-measure

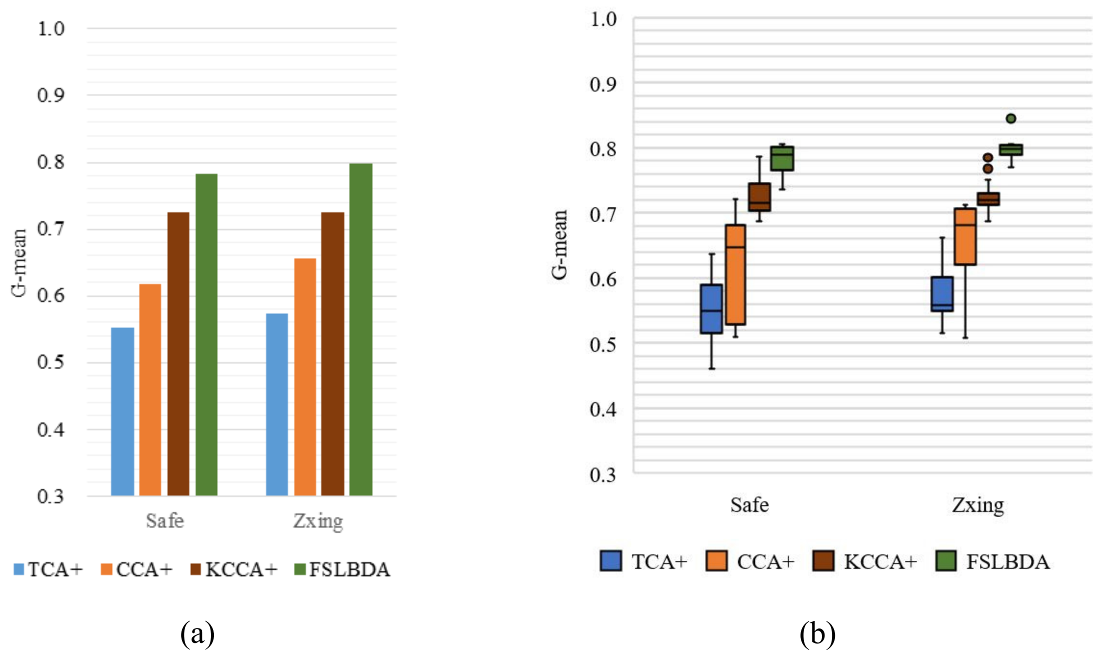


FIGURE 13. The G-mean of different approaches. (a) is the histogram, and (b) is the boxplot.

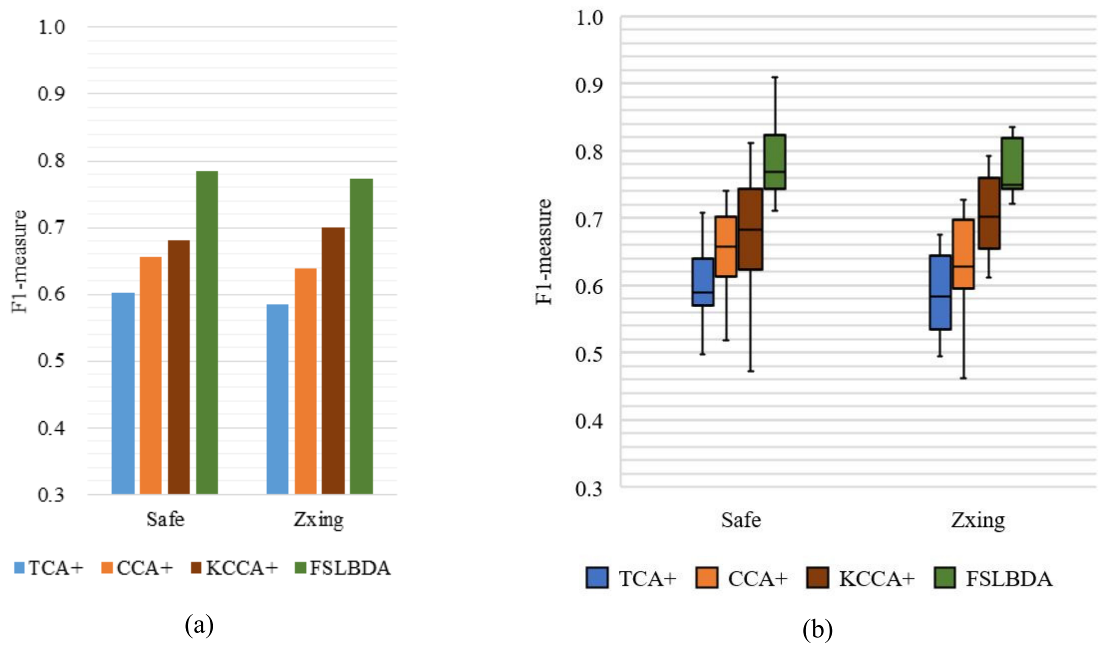


FIGURE 14. The F1-measure of different approaches. (a) is the histogram, and (b) is the boxplot.

generated by Safe and Zxing as source data. It can be found that AUC, G-mean, F1-measure of FSLBDA are better than other three approaches and increase largely.

Experimental results show that the prediction performance of FSLBDA proposed in this paper is better than other approaches. FSLBDA can better reduce the data difference between the source domain and the target domain to improve the prediction performance, especially for the prediction of imbalanced datasets.

The non-parametric test does not assume that the population distribution must conform to the normal distribution. It can infer that the population distribution directly from samples. The Kruskal-Wallis test is carried out under significance level $\alpha = 0.05$, and TCA+, CCA+, KCCA+, and FSLBDA are compared in pairs. The null hypothesis for each row in Table 4 show that the Method 1 and Method 2 distributions are the same. In order to reveal which of these groups differ from each other, we conduct a post hoc test

TABLE 3. Three indicator values of approaches when Zxing project as the source domain.

	AUC				G-mean				F1-measure			
Project	TCA+	CCA+	KCCA+	FSLBDA	TCA+	CCA+	KCCA+	FSLBDA	TCA+	CCA+	KCCA+	FSLBDA
PC1	0.598	0.746	0.800	0.852	0.517	0.713	0.702	0.805	0.524	0.596	0.760	0.744
PC3	0.577	0.732	0.799	0.889	0.558	0.671	0.718	0.801	0.611	0.462	0.667	0.833
PC4	0.686	0.796	0.737	0.806	0.601	0.696	0.751	0.789	0.587	0.697	0.793	0.833
MW1	0.606	0.775	0.810	0.824	0.515	0.708	0.707	0.805	0.495	0.613	0.660	0.744
CM1	0.673	0.717	0.807	0.875	0.641	0.524	0.722	0.770	0.584	0.694	0.713	0.722
ML	0.648	0.739	0.704	0.815	0.661	0.701	0.714	0.781	0.568	0.694	0.709	0.797
PDE	0.663	0.723	0.790	0.806	0.580	0.706	0.712	0.798	0.654	0.728	0.693	0.813
EQ	0.613	0.735	0.749	0.867	0.558	0.704	0.731	0.803	0.524	0.718	0.764	0.819
JDT	0.591	0.729	0.815	0.833	0.530	0.708	0.720	0.775	0.644	0.714	0.708	0.744
LC	0.616	0.717	0.806	0.889	0.557	0.681	0.728	0.806	0.536	0.603	0.612	0.732
AR1	0.659	0.645	0.748	0.852	0.597	0.583	0.687	0.799	0.538	0.602	0.655	0.750
AR3	0.732	0.676	0.745	0.877	0.569	0.508	0.768	0.794	0.616	0.569	0.649	0.746
AR4	0.756	0.674	0.870	0.892	0.609	0.621	0.785	0.795	0.675	0.667	0.702	0.836
AR5	0.679	0.715	0.769	0.912	0.549	0.653	0.724	0.845	0.664	0.594	0.630	0.757
AR6	0.615	0.731	0.844	0.909	0.556	0.646	0.719	0.805	0.535	0.628	0.791	0.746
mean	0.647	0.723	0.786	0.860	0.573	0.655	0.726	0.798	0.584	0.639	0.700	0.774

TABLE 4. Kruskal-Wallis H and Holm-Bonferroni correction.

Method 1	Method 2	AUC		F1-measure		G-mean	
		p-value	Holm-Bonferroni Correction	p-value	Holm-Bonferroni Correction	p-value	Holm-Bonferroni Correction
TCA+	CCA+	0.012	0.071	0.038	0.229	0.019	0.117
TCA+	KCCA+	0.001	0.007	<0.001	0.001	0.042	0.253
TCA+	FLSBDA	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
CCA+	KCCA+	0.001	0.006	0.001	0.005	<0.001	0.001
CCA+	FLSBDA	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
KCCA+	FLSBDA	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

with the Holm-Bonferroni correction. We use SPSS software to obtain adjusted p-value, which is directly compared with 0.05, and the difference is considered statistically significant if it is less than 0.05. Table 4 clearly shows that there is a significant difference between FSLBDA and TCA+, CCA+, and KCCA+.

VI. CONCLUSIONS & FUTURE WORK

In this paper, we introduce BDA to dynamically narrow the gap between marginal distribution and conditional distribution differences of heterogeneous datasets with the balance factor.

Since the defect datasets have class imbalance attributes and there are redundant metrics, we use the twice ensemble learning to solve this problem. XGBoost is used to rank the importance of metrics, adding complexity to the objective function to avoid over-fitting. We obtain the balanced small sample dataset through undersampling the non-defective samples, and use AdaBoost to predict the target modules, thus avoiding the under-fitting of the classification model.

The experimental results showed that the proposed FSLBDA approach is feasible and yields promising results. HDP is very promising, because it permits potentially all heterogeneous datasets of software projects to be used for defect prediction on new projects or projects that lack defect data. Furthermore, it may not be limited to defect prediction. This technique may be applicable to all predictive approaches for software engineering problems. In the future work, we will explore the feasibility of building various prediction models using heterogeneous datasets.

REFERENCES

[1] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, "An ensemble oversampling model for class imbalance problem in software defect prediction," *IEEE Access*, vol. 6, pp. 24184–24195, 2018.

[2] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 874–896, Sep. 2018.

[3] Z. Li, X.-Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, "Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction," *Automated Softw. Eng.*, vol. 25, no. 2, pp. 201–245, Jun. 2018.

[4] Z. Li, X.-Y. Jing, and X. Zhu, "Heterogeneous fault prediction with cost-sensitive domain adaptation," *Softw. Test. Verification Rel.*, vol. 28, no. 1, p. e1658, Jun. 2018.

[5] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "On the multiple sources and privacy preservation issues for heterogeneous defect prediction," *IIEEE Trans. Softw. Eng.*, vol. 45, no. 4, pp. 391–411, Apr. 2019.

[6] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Softw. Eng.*, vol. 14, no. 5, pp. 540–578, Oct. 2009.

[7] X. Yu, Z. Y. Ma, and C. X. Ma, "FSCR: A feature selection method for software defect prediction," in *Proc. 29th Int. Conf. Softw. Eng. Knowl. Eng.*, 2017, pp. 351–356.

[8] H. Qing, L. Biwen, S. Beijun, and Y. Xia, "Cross-project software defect prediction using feature-based transfer learning," in *Proc. 7th Asia–Pacific Symp. Internetware*, 2015, pp. 74–82.

[9] Y. Ma, S. Zhu, Y. Chen, and J. Li, "Kernel CCA based transfer learning for software defect prediction," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 8, pp. 1903–1906, 2017.

[10] W. Wen, B. Zhang, X. Gu, and X. Ju, "An empirical study on combining source selection and transfer learning for cross-project defect prediction," in *Proc. IEEE 1st Int. Workshop Intell. Bug Fixing (IBF)*, Feb. 2019, pp. 29–38.

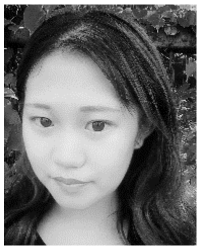
[11] J. Chen, Y. Yang, K. Hu, Q. Xuan, Y. Liu, and C. Yang, "Multiview transfer learning for software defect prediction," *IEEE Access*, vol. 7, pp. 8901–8916, 2019.

[12] Y. Liu, K. Liu, J. Yang, and Y. Yao, "Spatial-neighborhood manifold learning for nondestructive testing of defects in polymer composites," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2019.2949358.

[13] J. Chen, K. Hu, Y. Yang, Y. Liu, and Q. Xuan, "Collective transfer learning for defect prediction," *Neurocomputing*, to be published, doi: 10.1016/j.neucom.2018.12.091.
[14] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.
[15] S. Satpal and S. Sarawagi, "Domain adaptation of conditional probability models via feature subsetting," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Berlin, Germany: Springer, 2007, pp. 224–235.
[16] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2200–2207.
[17] C.-A. Hou, Y.-H.-H. Tsai, Y.-R. Yeh, and Y.-C.-F. Wang, "Unsupervised domain adaptation with label and structural consistency," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5552–5562, Dec. 2016.
[18] J. Tahmoresnezhad and S. Hashemi, "Visual domain adaptation via transfer feature learning," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 585–605, Feb. 2017.
[19] C.-A. Hou, Y.-R. Yeh, and Y.-C.-F. Wang, "An unsupervised domain adaptation approach for cross-domain visual classification," in *Proc. 12th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2015, pp. 1–6.
[20] J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen, "Balanced distribution adaptation for transfer learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 1129–1134.
[21] R. E. Schapire, "The strength of weak learnability," in *Proc. Annu. Symp. Found. Comput. Sci., Res.*, Triangle Park, NC, USA, 1989, pp. 28–33.
[22] M. Ren, W. Y. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. 35th Int. Conf. Mach. Learn. (PLMR)*, 2018, pp. 4334–4343.
[23] Z. Li, X.-Y. Jing, X. Zhu, and H. Zhang, "Heterogeneous defect prediction through multiple kernel learning and ensemble learning," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2017, pp. 91–102.
[24] Z. Q. Li, X. Y. Jing, X. K. Zhu, H. Y. Zhang, B. W. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Softw. Eng.*, vol. 26, no. 3, pp. 599–651, Jun. 2019.
[25] R. X. Wu and H. Y. Zhang, "ReLink: Recovering links between bugs and changes," in *Proc. ACM SIGSOFT Symp. Found. Softw. Eng.*, Szeged, Hungary, 2011, pp. 5–9.
[26] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *Proc. 7th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, May 2010, pp. 31–41.
[27] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IIEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.
[28] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 382–391.
[29] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," in *Proc. 10th Joint Meeting Found. Softw. Eng. (ESEC/FSE)*, 2015, pp. 496–507.

AILI WANG was born in Tianjin, China, in 1979. She received the B.S., M.S., and Ph.D. degrees in information and signal processing from the Harbin Institute of Technology, Harbin, China, in 2002, 2004, and 2008, respectively.

She joined the Harbin University of Science and Technology as an Assistant, in 2004, and became an Associate Professor and a Master Tutor with the Department of Communication Engineering, in 2010. She has been a Visiting Professor to do the research of 3D polyp reconstruction with the Computer Science Laboratory, Chubu University, Japan, in 2014. She is the author of two books, more than 80 articles which are published on the IEEE conferences and journals (EI indexed or SCI indexed). She is the Chairman of 11th EAI International on Wireless and Satellites (WISATS) and 7th EAI International Conference on Green Energy and Networking (GreeNets). Her research interests include image super resolution, image fusion, and object tracking.



YUTONG ZHANG received the bachelor's degree from the College of Electronic and Information Engineering, Heilongjiang University of Science and Technology, in 2017. She is currently pursuing the master's degree with the College of Communication and Information System, Harbin University of Science and Technology. Her research interests include data mining and applications, and software engineering.

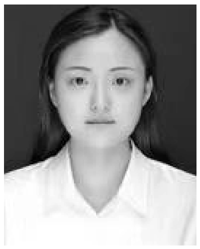
HAIBIN WU was born in Harbin, China, in 1977. He received the B.S. and M.S. degrees from the Harbin Institute of Technology, Harbin, China, in 2000 and 2002, respectively, and the Ph.D. degree in measuring and testing technologies and instruments from the Harbin University of Science and Technology, Harbin, China, in 2008.

From 2009 to 2012, he held a postdoctoral position with the Key Laboratory of Underwater Robot, Harbin Engineering University. From 2014 to 2015, he was a Visiting Scholar with the Robot Perception and Action Laboratory, University of South Florida. Since 2012, he has been a Professor with the Instrument Science and Technology Discipline, Harbin University of Science and Technology. He is the author of three books, more than 40 articles, and more than 20 inventions. His research interests include robotic vision, visual measuring and image processing, medical virtual reality, and photoelectric testing.

Dr. Wu is an Editorial Board Member of the *Chinese Journal of Liquid Crystals and Displays*. He was the Director of the Precision Machinery Branch, China Instrumentation Society, and the Director of Visual Inspection Committee of Chinese Graphic and Image Society. He is an Associate Editor-in-Chief of the *Journal of Harbin University of Science and Technology*.

KAIYUAN JIANG was born in Harbin, China, in 1982. He received the B.S. and M.S. degrees from Xidian University, Xi'an, China, in 2006 and 2009, respectively, and the Ph.D. degree in communication and information system from the Harbin Institute of Technology, Harbin, in 2014.

He joined the Harbin University of Science and Technology, in 2014. He is currently a Postgraduate Tutor with the College of Measurement and Control Technology and Communication Engineering. His research work mainly focuses on image fusion, class imbalance learning, and wireless network simulation.



MINHUI WANG was born in Linkou, Mudanjiang, Heilongjiang, China, in 1995. She received the bachelor's degree in aerospace engineering from the Harbin University of Science and Technology, in 2017, where she is currently pursuing the master's degree in electronics and communication engineering. She has published a conference paper and a journal article in the field of LiDAR research (EI indexed or SCI indexed). Her research direction is classification of LiDAR.